Combinatorial Optimization Algorithms for Radio Network Planning

Patrice Calégari, Frédéric Guidec, Pierre Kuonen

Swiss Federal Institute of Technology, 1015 Lausanne, Switzerland.
E-mail: {calegari,guidec,kuonen}@di.epfl.ch

Frank Nielsen [1]

École Polytechnique, Laboratoire d'informatique LIX, CNRS Unité 1439
91128 Palaiseau Cedex, France.

Abstract

This paper uses a realistic problem taken from the telecommunication world as the basis for comparing different combinatorial optimization algorithms. The problem recalls the minimum hitting set problem, and is solved with greedy-like, Darwinism and genetic algorithms. These three paradigms are described and analyzed with emphasis on the Darwinism approach, which is based on the computation of ϵ-nets.

Key words: Combinatorial optimization, Radio transceiver siting, Set system, Genetic algorithm, Parallel computing.

1 Introduction

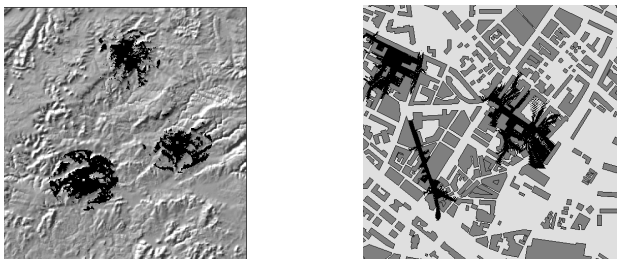
One of the key issues telecommunication companies must face when deploying a mobile phone network is the selection of a good set of sites among those possible for installing *Base Transceiver Stations* (BTSs). The problem comes down to serving a maximum surface of a geographical area with a minimum number of BTSs. The set of sites where BTSs may be installed is taken as an input, and our goal is to find a minimum subset of sites that allows a 'good' service. This *mobile radio network planning* problem is tackled in the

[1] The author is now a research associate of SONY Computer Science Laboratories Inc., Takanawa Muse Bldg., 3-14-13 Higashi Gotanda, Shinagawa-Ku, Tokyo 141-0022, Japan. E-mail: `Nielsen@csl.sony.co.jp`

STORMS project (Software Tools for the Optimization of Resources in Mobile Systems). We have developed so far different algorithms to solve this *mobile radio network planning* problem.

The paper is organized as follows: Section 2 introduces notations and gives a model of our problem. Section 3 describes a greedy-like approach. Section 4 briefly introduces ϵ-nets and some of their key properties. ϵ-nets are then applied to solve the problem in a weighted process. Section 5 gives an island-based genetic algorithm that runs concurrently on a network of workstations. Finally, we elaborate on comparative benchmarks obtained experimentally when siting transceivers in the Swiss urban district of Geneva and in the hilly French region 'Les Vosges'.

2 Modeling of the problem



(a) (b)

Fig. 1. Three cells computed on the French region 'Les Vosges' (a), and in a district of the city of Geneva (b). The black zone represents the served area.

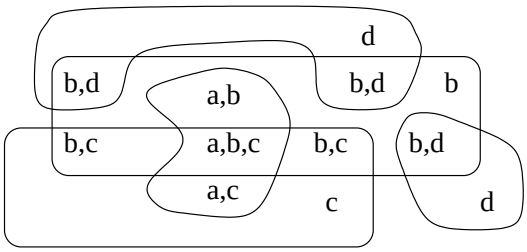
Cells. A geographical location is said to be *served* when it can receive the signal broadcast by a BTS with a given quality of service. The area served by a BTS is called a *cell*. It must be noticed that, since each BTS is associated to a cell, we will not differentiate between BTSs and cells in the remainder of this paper. In our implementation, geographical locations are discretized on a *regular grid*, and the cells are computed by a radio wave propagation prediction tool. Figure 1 shows the shape of such cells, computed in the hilly French region 'Les Vosges' and in the Swiss urban district of Geneva (in this example, indoor radio wave propagation is not considered).

Modeling of the service. The relationship between each pixelized location served and the BTSs is naturally modeled as a bipartite graph whose nodes represent either BTSs or geographic locations (pixels). Such a graph tends

to have huge size when many geographic locations are allowed. A smart way to reduce the graph size without loosing any useful information is to build a bipartite graph whose nodes represent either BTSs or *intercells*. An intercell is defined as the set of geographical locations that are potentially served by exactly the same set of BTSs. For each intercell node, one only needs to encode the *cost* of this intercell, that is, the number of locations it contains. It can be noticed that the geographical zone delimited by an intercell node is not necessarily connex. The bipartite graph hence obtained can be smaller than the former one by more than one order of magnitude.

Below, we introduce formal definitions used in the field of combinatorial optimization theory and explain their equivalence or relationships with our facility location problem.

Set system. A *set system* $(X, \Re)$ is a set X of n *elements*, with a collection $\Re$ of m subsets of X called in the literature *ranges*. Let us consider a set system $(X, \Re)$ where X is the set of cells and $\Re$ is the set of all *intercells*. Figure 2 depicts such a set system.



$$X = \{a, b, c, d\}$$

$$\Re = \{\{b\}, \{c\}, \{d\}, \{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{a, b, c\}\}$$

Fig. 2. A set system of $n = 4$ cells inducing $m = 8$ intercells.

Hitting set and set cover problems. A *hitting set* of our set system $(X, \Re)$ is a subset $H \subseteq X$ of cells such that H has a non-empty intersection with every intercell R in $\Re$ (for example, in Figure 2 $\{b, c, d\}$ is a hitting set of $(X, \Re)$). Roughly speaking, this means that each pixelized location of the whole area is served by at least one BTS. The problem we consider recalls the minimum *Hitting Set Problem* (HSP) whose NP-completeness (shown by Karp [8]) dates back to the early seventies. However, it slightly differs from minimum HSP because our goal is to select a satisfactory subset of BTSs that ensures a service in *almost* all the area. This means that non-combinatorial parameters such as the target service ratio ($tsr \in [0, 1]$) are also to be taken into account in practice. tsr expresses the ratio of the area that is targeted to be served over the maximum area that can be served. For each cell $x \in X$, let $x_\Re = \{R \in \Re | x \in R\}$ denote the set of intercells included in x. Let $X_\Re = \{x_\Re | x \in X\}$ be the set of groups of all intercells. A *covering set* of a set system $(\Re, X_\Re)$ is a subset of $X'_\Re \subseteq X_\Re$ such that $\cup_{x_\Re \in X'_\Re} = \Re$. A *minimum set cover problem*, SCP, asks for a minimum-size covering set $X_{\Re min}$ such that

$|X_{\Re min}| = \min(|X'_{\Re}|, X'_{\Re} \subseteq X_{\Re}$ and $\cup_{x_{\Re} \in X'_{\Re}} = \Re)$. The set system $(\Re, X_{\Re})$ is said to be the *dual* of $(X, \Re)$ since a solution to HSP implies a solution to SCP and vice-versa. Denote by k-HSP the HSP where each range has at most k BTS and by k-SCP the SCP where each cell covers at most k intercells. Recently, Feige [6] proved that unless $NP \subseteq DTIME[n^{\log\log n}]$ there is no polynomial-time algorithm that guarantees a $(1-\epsilon)\log k$ *performance ratio*.[2] This plainly explains the intractability of HSP, and dually of SCP, from both the theoretical and practical point of view and motivates our comparative tests. The following section presents the greedy algorithm for solving SCP, refines the modeling of the problem and gives further references to extensions of SCP.

3 A greedy-like algorithm

Introduced by Chvátal [4] and thoroughly analyzed by Slavík [15], the natural greedy heuristic surprisingly achieves a performance ratio of $\log k - \log\log k + \Theta(1)$, where k is the maximum number of intercells lying in a BTS. Note that if $k \leq 2$ then SCP is equivalent to the EDGE COVER problem and therefore can be solved in $O(n\sqrt{m})$-time using a *maximum matching* in a bipartite graph [12]. This approach was used to improve GREEDY and k-HSP [5]. The PARTIAL HITTING SET COVER PROBLEM, PHSP, consists in hitting, with as few cells as possible, at least $|\Re|r$ intercells for a given hitting ratio $r \in [0,1]$. PHSP has also been proven NP-complete by Kearns [9] as soon as $0 < r \leq 1$. Kearns used a variant of GREEDY with performance ratio $2H(m)+3$, where $H(c) = \sum_{i=1}^{c} \frac{1}{i} \leq \log c + 1$ is the c^{th} *harmonic number*. Slavík [14] lowered the performance ratio to $\min\{H(\lceil rm \rceil), H(k)\}$. A *weighted set system* is a set system in which each range $R \in \Re$ is given a cost c_R (e.g., c_R is the number of pixels contained by the intercell R). Denote $c(x_{\Re}) = \sum_{R \in x_{\Re}} c_R$ and $c(X) = \sum_{x \in X} c(x)$. GREEDY can be naturally extended to weighted set systems and runs in $O(nm)$ time and space for dense (resp. $O(n\log n)$ for sparse, i.e. $m = O(n)$) set systems as shown below (See algorithm GREEDY). There exists various extensions of HSP that lead to different heuristics and hardness results. We refer to [13] for an up-to-date survey.

[2] the performance ratio is the ratio between a solution and an optimal solution.

Algorithm GREEDY *// Implements Kearn's greedy-like heuristic*
// tsr is the target service ratio
// rest is the surface yet to be served in order to obtain a tsr-service
$X' := \emptyset$; $rest := tsr$
// Initialize the current solution X' with 0 BTS
while less than tsr of the surface is served by X' **do**
  Add x_i (the i^{th} BTS) to X' such that x_i maximizes $\min(rest, c(x_{i\Re} \backslash X'_{\Re}))$
  Update $rest := tsr - \sum_{x \in X'} c(x)$

4 A Darwinism algorithm

4.1 Definition and properties of epsilon-nets

If a subset $N \subseteq X$ intersects each set R of $\Re$ of size bigger than $\epsilon * |X|$, then N is called an *ϵ-net*. In our case, an ϵ-net is a set of cells (i.e., BTSs) that serves all intercells potentially served by more than $\epsilon * n$ BTSs. Denote by $\Re_{|Y}$ the collections of intercells restricted to elements of Y: $\Re_{|Y} = \{R \cap Y | R \in \Re\}$. A set $Y \subseteq X$ is said *shattered* if $\Re_{|Y} = 2^Y$, where 2^Y denotes all subsets of Y. The *Vapnik-Červonenkis dimension* is defined as the minimum integer d so that no $d+1$ elements can be shattered by $\Re$. Set systems of VC-dimension d admits $\frac{1}{r}$-nets of size $O(dr \log r)$ [10] and can be computed in $O(d)^3 D r^D \log^D(rd)|X|$ given a computational oracle that returns $(Y, \Re_{|Y})$ in time $O(|Y|^{D+1})$ [11].

4.2 Weighted set algorithm

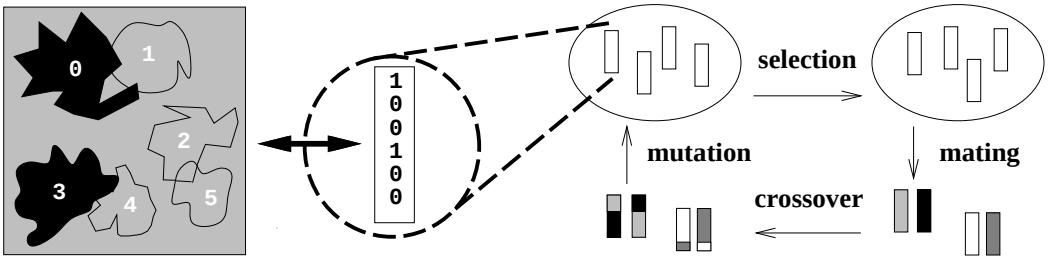
In our practical situation, the VC-dimension of the problem can be bounded by 5 since it is very unlikely to obtain a complete sub-arrangement of 5 BTSs (i.e., all proper $2^5 - 1$ intercells generated by 5 BTSs). Algorithm WEIGHTED_SETS is based on seminal ideas of a weighted strategy presented in [1]. Parameters α and β are used for both convergence of the algorithm and performance ratio of our solution. Basically, the strategy amounts to guess the optimal size c and compute an ϵ-net which might be a 'good' solution or not. In the latter case, we choose a not yet covered intercell, update the weights of all BTSs fully serving it and reiterate until, at some step, we find a 'good' solution.

5 An island-based genetic algorithm

The algorithm presented in the previous section relies on the notion of ϵ-net and set systems. In this section we present a genetic algorithm which does

Algorithm EPSILON_NET(ϵ) // *Computes randomly an ϵ-net.*
$i := \lceil \frac{1}{\epsilon} \rceil$; $Q := \emptyset$
while Q is not an ϵ-net **do**
  $Q := \emptyset$
  Draw i elements of X taking into account their weight
  $i := i + 1$

Algorithm WEIGHTED_SETS // *Implements a weighted selection procedure*
// *n number of initial potential BTSs, i.e.* $100 < n < 10000$
// *$[c_{min}, c_{max}] \subseteq [0, n]$ potential interval where a solution lies*
// *tsr is the target service ratio (tsr* $\simeq 0.9$*)*
// *α, β: we must have $\alpha > \frac{\beta - 1}{\ln \beta}$ to prove the convergence (e.g., $\beta \simeq 2$, $\alpha \simeq 1.45$)*
for $c \in [c_{min}, c_{max}]$
  // *c is supposed to be the optimal value*
  Set $w_i := 1$ be the weight of $x_i \in X$.
  repeat at most $\frac{\alpha c \ln \frac{n}{c}}{\alpha \ln \beta - \beta + 1}$ times
    Compute X' a weighted $\frac{1}{\alpha c}$-net of $(X, \Re)$
    if X' serves more than *tsr* of the surface
      then
        X' is a 'good' enough solution. **STOP**.
      else
        Choose a not yet served intercell $R \in \Re$
        Multiply by β ALL the weights w_i's of the BTSs that can serve it.



(a) (b) (c)

Fig. 3. Two sites (associated to cells in black) from a set of six potential ones are chosen in this candidate solution (a), that can be encoded as a bit-string (b). A set (or *population*) of such bit-strings (or *individuals*) evolves according to the four phases of a genetic algorithm (c).

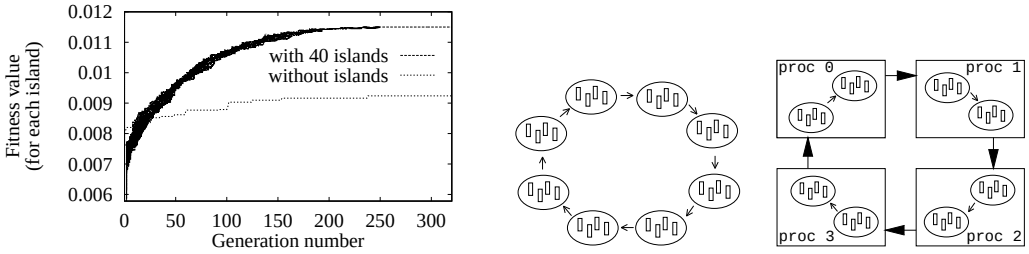
not rely on these notions but that still uses the bipartite graph introduced in Section 2 to compute the service ratio.

A genetic algorithm is a population-based algorithm, which means that its state at any time is a set of candidate solutions, called a *population* of *individuals*. In our implementation, an individual is encoded as a chromosome-like bit string that represents the whole set of possible BTS sites. Whether a lo-

cation is actually selected in a potential solution depends on the value of the corresponding entry in the bit string (see Figure 3.a and 3.b). A fitness value is assigned to each individual, indicating how 'good' the solution it represents is. We choosed $fitness = \frac{(\text{served area/total area})^{\alpha}}{\text{number of BTSs used}}$, where α is a parameter that can be tuned at will (experiments show that when $\alpha = 4$ the solutions returned by the algorithm give around 90% of service ratio, which is a satisfactory result according to telecommunication specialists).

In genetic algorithms (introduced by Holland in [7]) four phases inspired by the genetic mechanisms of natural species evolution can be identified (see Figure 3.c). Given a population of individuals, that are initially generated at random, an intermediate population is created by selecting individuals according to their relative fitness value: the higher the fitness value of an individual, the more likely it is to be selected. When this intermediate population has been filled, individuals are mated in pairs. On each of these pairs, a *1-point crossover* operator is applied with a probability of p_c. This operator cuts two given bit strings at a same random position and recombines them by exchanging their ends, thus producing two new bit strings (or *offsprings*). In the end, a *mutation* operator is applied to each individual with probability p_m. The mutation operator inverts the value of a randomly selected bit of the bit-string. This introduces 'noise' to prevent premature convergence of the population. The execution terminates after a predefined number of generations (typically twice the total number of individuals).

Such a genetic algorithm meets our demand. Yet, it has two major shortcomings. First, it is not fast enough for interactive use, as required by telecommunication operators. Second, the solutions obtained remain far from the optimum solution because of an over rapid convergence of the algorithm.



(a) (b) (c)

Fig. 4. Comparison of the convergence speed, with and without islands, of a genetic algorithm operating on 160 individuals (a). Islands (or sub-populations) of a genetic algorithm organized according to a ring topology (b). Distribution of islands in a parallel version of the algorithm (c).

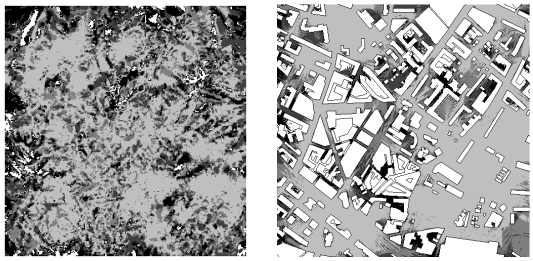
Many studies have been done to improve the quality of the results obtained with genetic algorithms [17]. One of them consists in splitting the population

in subpopulations, called islands, that evolve independently [16], and that can cooperate by *migrating* individuals from an island to another. An example of convergence speed observation, with and without islands, is shown in Figure 4.a (a more detailed study can be found in [2,3]).

A population must contain many individuals in order to give good results, hence a large computation load. The amount of independent processing required for the evolution of islands suggests an intrinsic parallelism. The overall computation time could thus be decreased by distributing the islands on several processors. In our implementation, the islands are virtually positioned on an oriented ring, and migrations are only allowed along that ring (see Figure 4.b). Every time a new generation is computed, a copy of the best individual (that with the greatest fitness value) ever met by each island is sent to the next island on the ring. Each island thus receives a new individual that replaces one of its individuals selected randomly. This topology was chosen so as to minimize the amount of migrations, and thus to minimize the communication load due to migrations between remote islands (see Figure 4.c).

6 Analysis of the results

6.1 Quality

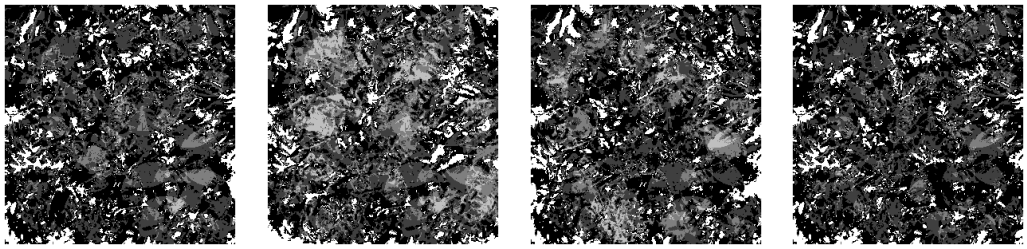


(a) (b)

Fig. 5. Served areas computed on the French region 'Les Vosges' with 150 BTSs (a), and in a district of the city of Geneva with 99 BTSs (b). White zones represent areas that are not served. Black zones are served once. Dark grey zones are served twice, and light grey zones are served three times or more. The underlying terrain is not represented on these pictures.

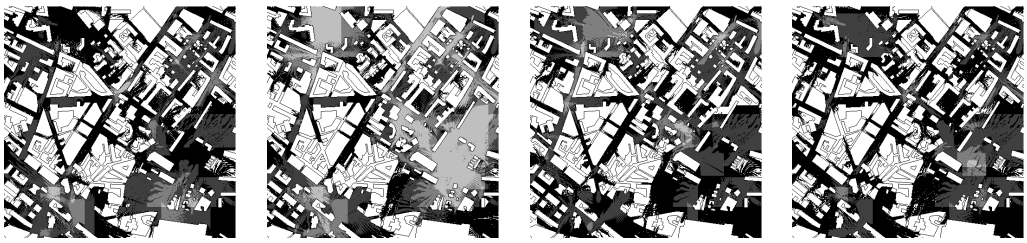
For our tests, a rural and an urban real-life case are considered. First, a set of 150 potential sites is considered in the French eastern hilly region 'Les Vosges'. Second, a set of 99 potential sites is considered in a district of the Swiss city of Geneva. Figures 5 shows the total coverage that would be obtained in both cases if all the potential sites are selected to install BTSs.

GREEDY gives quite good results in the two cases and does not need to be tuned. The parameters of the WEIGHTED_SETS algorithm are set such that $\alpha = 1.45, \beta = 2$ and $c_{min} = 15$. The genetic algorithm runs with 160 individuals that evolve during 320 generations. Experience shows that the probabilities of mutation and crossover give better results when they are high (typically $p_m, p_c \in [0.6, 0.9]$). The island-based algorithm distributes the 160 individuals on 40 islands.



(a) (b) (c) (d)

Fig. 6. Results obtained with an initial set of 150 BTSs in the French region 'Les Vosges', taken as an input by our different algorithms. The target service ratio is 90%. The greedy-like algorithm returns a solution with 58 BTSs (a). The Darwinism algorithm returns a solution with 82 BTSs (b). The genetic algorithm returns a solution with 70 BTSs (c), and the island-based genetic algorithm returns a solution with 57 BTSs (d). The meaning of the grey scale is the same as that of Figure 5.



(a) (b) (c) (d)

Fig. 7. Results obtained with an initial set of 99 BTSs in a district of the city of Geneva, taken as an input by our different algorithms. The target service ratio is 90%. The greedy-like algorithm returns a solution with 24 BTSs (a). The Darwinism algorithm returns a solution with 38 BTSs (b). The genetic algorithm returns a solution with 30 BTSs (c), and the island-based genetic algorithm returns a solution with 22 BTSs (d). The meaning of the grey scale is the same as that of Figure 5.

Figure 6 and 7 shows examples of solutions that are found by our algorithms. It can be noticed that the number of locations that are covered more than once are very small. This side effect is due to the fact that the algorithms tend to minimize the overlaps between cells.

Figure 8 shows the characteristics of solutions that are obtained by our differ-

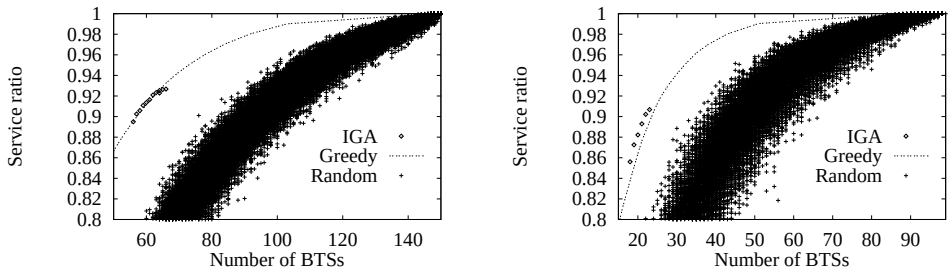


Fig. 8. Service ratio against the number of BTSs selected by the Island-based Genetic Algorithm (IGA), by GREEDY, and at random. The input data is the 150 BTS set of 'Les Vosges' and the 99 BTS set of the Geneva district introduced above.

ent programs. Before comparing the quality of the solutions, we first compute a set of randomized solutions to serve as a basis for comparison. The different solutions obtained by the island-based genetic algorithm are due to several runs with different random seeds. Since at each step of GREEDY, a partial solution exists, the evolution of the service ratio can be observed for any number of BTSs selected.

The results of the island-based genetic algorithm and of GREEDY are of the same quality in average. However, we do not know how far these results are from the optimal solution. Experience shows that when an optimal solution is known, it can be found by the island-based genetic algorithm whereas GREEDY can fall in bad, yet attractive local optima.

So far, the WEIGHTED_SETS algorithm shows rather poor quality results. Its solutions are only slightly better than the best of those generated randomly. The program must indeed be improved and fine tuned. Beside it should be tested on data sets larger than those shown in this paper (thousand of BTSs). Actually, since this algorithm is based on probabilistic rules, hundreds of BTSs may not be sufficient to let it work properly. Another clue for explaining these poor quality results is the very poor performance of the EPSILON_NET algorithm introduced in this paper which is the core of the WEIGHTED_SETS algorithm. A better approach using a randomized greedy-like or a weighted greedy-like algorithm is being investigated and seems to show results of better quality.

6.2 Execution times

GREEDY shows the best ratio $\frac{\text{quality}}{\text{time}}$. However slightly better solutions can be found by the island-based genetic algorithm.

Algorithm	Execution time	
	Vosges	Geneva
Greedy algorithm	3.6 sec.	0.6 sec.
Darwinism algorithm	$\approx$ 2 min.	$\approx$ 2 min.
Genetic algorithm (sequential)	24 min.	16 min. 19 sec.
Island-based genetic algorithm (40 islands in sequential)	4 min. 30 sec.	3 min. 31 sec.
Island-based genetic algorithm (40 islands on 40 workstations in parallel)	12 sec.	10 sec.

Table 1
Execution times of different algorithms applied on an initial set of 150 BTSs in the French region 'Les Vosges', and on a set of 99 BTSs in a district of Geneva.

The number of operations achieved during the selection step of the genetic algorithm is proportional to the square of the number of individuals per island. Actually, when the number of islands is doubled, the number of individuals is divided by 2 on each island and the computation load of the selection step is divided by $2^2 = 4$. This explains why the execution using 40 islands is much quicker than that evolving a single island.

When running the island-based genetic algorithm in parallel, a speedup of up to 7.8 was observed on 10 Sparc-4 workstations (that is, the same execution of the program runs 7.8 times quicker on 10 machines than on a single one). The resulting efficiency of 78% is considered as very good since communications between remote processors are usually much time consuming in parallel programs. However, this efficiency falls down to 37% on 86 workstations. For this latter result, 320 individuals were distributed on 160 islands.

7 Conclusion and future works

In this paper, we show three very different approaches to solve the minimum set cover problem and the minimum hitting set problem. A mobile radio network planning project that deals with this problem is used to test these three algorithms. The results obtained with a classical greedy-like algorithm and an island-based genetic algorithm are satisfying. Moreover the computation time needed by the genetic algorithm, that gives the best results, can be decreased by running a parallel version of the program. The original Darwinism WEIGHTED_SETS algorithm must still be improved and tested, but it shows new perspectives for the ϵ-net theory.

The resulting *C++* package currently weighs about $50,000$ code lines and its object-oriented conception allows to easily implement other algorithms such as Evolution Strategy, ant systems and tabu search. The implementation of a hybrid method that combines different algorithms is already in progress.

Acknowledgements

The STORMS project is a European ACTS project, funded by the European Community and by the Swiss government (OFES grant). The library APPEAL (Advanced Parallel Population-based Evolutionary Algorithm Library) developed in the project LEOPARD (parallel population-based methods for combinatorial optimization) was used to implement our island-based genetic algorithm. The LEOPARD project is funded by the Swiss National Science Fund (# 21-45070.95/1).

References

[1] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete and Computational Geometry*, 14:263–279, 1995.

[2] P. Calégari, F. Guidec, P. Kuonen, and D. Kobler. Parallel Island-Based Genetic Algorithm for Radio Network Design. *Journal of Parallel and Distributed Computing (JPDC): special issue on Parallel Evolutionary Computing, Academic Press*, 47(1):86–90, Nov. 1997.

[3] P. Calégari, P. Kuonen, F. Guidec, and D. Wagner. A Genetic Approach to Radio Network Optimization for Mobile Systems. In *Proceedings of the IEEE 47th Vehicular Technology Conference (VTC)*, volume 2, pages 755–759, May 1997.

[4] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematical Operations Research*, 3:233–235, 1979.

[5] R. Duh and M. Fürer. Approximation of k-set cover by semi-local optimization. In *Proceedings of the 29th Annual ACM Symposium on Theory Computation*, pages 256–264, 1997.

[6] U. Feige. A Threshold of $\log n$ for Approximating Set Cover. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, 1996.

[7] J. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, 1975.

[8] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103, New York, 1972. Plenum Press.

[9] M. J. Kearns. *The Computational Complexity Problems.* MIT Press, 1990.

[10] J. Komlós, J. Pach, and G. Woeginger. Almost tight bounds for ϵ-nets. *Discrete and Computational Geometry*, 7:163–173, 1992.

[11] J. Matoušek. Epsilon-nets and computational geometry. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics*, pages 69–89. Springer-Verlag, 1993.

[12] S. Micali and V. V. Vazirani. An $O(|E|\sqrt{|V|})$ algorithm for maximum matching in general graphs. In *Proceedings 21st IEEE Annual Symposium on the Foundations of Computer Science*, pages 17–27, 1980.

[13] V. Paschos. A survey on approximately optimal solution to some covering and packing problems. *ACM Computing Surveys*, 29(2):172–209, June 1997.

[14] P. Slavík. Improved Performance of the Greedy Algorithm for Partial Cover. *Information Processing Letters*, 64(5):251–254, 15 Dec. 1997.

[15] P. Slavík. A Tight Analysis of the Greedy Algorithm for Set Cover. *Journal of Algorithms*, 25(2):237–254, Nov. 1997.

[16] R. Tanese. Distributed Genetic Algorithms. In J. Schaffer, editor, *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 434–439, George Mason University, June 1989. Morgan Kaufmann.

[17] D. Whitley. A genetic algorithm tutorial. Technical Report CS-93-103, Colorado State University, 1993.